The University of Jordan

Authorization Form

I, Radwan Mohamed Radwan AL-Shalalfa, authorize the University of Jordan to supply copies of my Thesis to libraries or establishments or individuals on request, according to the University of Jordan regulations.

Signature: رضوان

Date: 16/8/2011

August, 2011

الجامعة الأردنيــة
كلية الدراسات العليا

التاريــــخ: / /

نموذج رقم (١٨)
اقرار والتزام بقوانين الجامعة الأردنية وأنظمتها وتعليماتها لطلبة الماجستير

أنا الطالب: رضوان محمد رضوان الشوابكة الرقم الجامعي: ٨٠٦٠٤٠٦
التخصص: ماجستير علم الحاسوب الكلية: تكنولوجيا المعلومات

اعلن بأنني قد التزمت بقوانين الجامعة الأردنية وأنظمتها وتعليماتها وقراراتها السارية المفعول المتعلقة باعداد رسائل الماجستير والدكتوراة عندما قمت شخصيا" باعداد رسالتي / اطروحتي بعنوان: ....
Best uTilization of mirroring server Resources using Artificial neural networks

وذلك بما ينسجم مع الأمانة العلمية المتعارف عليها في كتابة الرسائل والأطاريح العلمية. كما أنني أعلن بأن رسالتي /اطروحتي هذه غير منقولة أو مستلة من رسائل أو أطاريح أو كتب أو أبحاث أو أي منشورات علمية تم نشرها أو تخزينها في أي وسيلة اعلامية، وتأسيسا" على ما تقدم فانني أتحمل المسؤولية بأنواعها كافة فيما لو تبين غير ذلك بما فيه حق مجلس العمداء في الجامعة الأردنية بالغاء قرار منحي الدرجة العلمية التي حصلت عليها وسحب شهادة التخرج مني بعد صدورها دون أن يكون لي أي حق في التظلم أو الاعتراض أو الطعن بأي صورة كانت في القرار الصادر عن مجلس العمداء بهذا الصدد.

توقيع الطالب: .... التاريخ: ١٦/ ٨ / ٢٠١١

BEST UTILIZATION OF MIRRORING SERVER RESOURCES USING ARTIFICIAL NEURAL NETWORKS

By

Radwan Mohamed Radwan Al-Shalalfa

Supervisor
Dr. Mousa Tawfiq AL-Akhras

Co-Supervisor
Dr. Azzam Sleit

This Thesis was Submitted in Partial Fulfillment of the Requirements for the Master's Degree of Computer Science

Faculty of Graduate Studies

The University of Jordan

August, 2011

COMMITTEE DECISION

This Thesis (BEST UTILIZATION OF MIRRORING SERVER RESOURCES USING ARTIFICIAL NEURAL NETWORKS) was Successfully Defended and Approved on August, 2011

<u>Examination Committee Signature</u>

Dr. Mousa Al Akhras. (Supervisor)
Assoc. Prof. of Artificial Neural Networks and communications

Dr. Azzam Sleit. (Co- Supervisor)
Assoc. Prof. of Information Retrieval

Dr. Abdul-Latif Abu Dalhoum. (Member)
Assoc. Prof. of Genetic Algorithms

Dr. Iman Musa Al-Momani (Member)
Assist. Prof. of Wireless Networks and Security

Dr. Mohammad A.Abbadi (Member)
Assoc. Prof. of Image processing
(Mutah University)

الجامعة الأردنيــة
كلية الدراسات العليا

التاريخ: / /

نموذج رقم (27)
تسليم رسالة ماجستير جامعية للمكتبة

الدكتور مدير المكتبــة

تحية طيبة وبعد،،،

لقد ناقش الطالب / الطالبة: رضوان محمد رضوان الشلالفة.................. ورقمه الجامعي: 8060456

تخصص الماجستير: علم حاسوب..

يوم: الاربعاء.......................... الموافق: 10 / 08 / 2011 ، وكانت النتيجة ناجحاً.

عنوان الرسالة (باللغة التي كتبت بها الرسالة)

BEST UTILIZATION OF MIRRORING SERVER RESOURCES USING ARTIFICIAL NEURAL NETWORKS

نرجو استلام النسخة الورقية التي تمت الموافقة عليها في صيغتها النهائية من قبل المشرف ولجنة المناقشة، ونسخة من الرسالة على القرص المضغوط (CD)، وذلك لإيداعها في المكتبة حسب الأصول.

وتفضلوا بقبول فائق الاحترام،،،

المشرف	رئيس قسم التخصص أو نائب رئيس لجنة الدراسات العليا في كلية التخصص	نائب عميد كلية الدراسات العليا
د. موسى توفيق الأحمر		
التوقيع..............	التوقيع..............	التوقيع..............
التاريخ: 2011/8/15	التاريخ: ٢٦/ ٨/ ٢٠١١	التاريخ:

مواصفات الاقراص المدمجة الخاصة بالرسائل الجامعية

- ان يضم القرص المدمج كافة المعلومات الواردة في النسخة الورقيه من الرسالة وذلك ضمن ملف واحد.
- ان يكون ترتيب الرسالة على القرص حسب ترتيب النسخة المطبوعة ورقيا.
- ان يحتوي القرص على صورة (save as jpg) عن اجازة الرسالة موقعة وموثقة من اعضاء لجنة المناقشة ومعتمدة من قبل الجامعة.
- تخزين الرسالة في ملف آخر على شكل (Acrobat reader PDF) لتسهيل تفعيل الرسالة على شبكة الانترنت ضمن قاعدة الرسائل الجامعية كاملة النص.

علما" أنه لن يكون بالامكان توثيق أي رسالة غير مطابقة للمواصفات المذكورة أعلاه.

DEDICATION

First of all, I would like to thank the Almighty ALLAH for helping and supporting me to finish this work, to my beloved parents for their support and encouragement, to whoever taught me a letter on my way of knowledge.

ACKNOWLEDGMENT

Special thanks to my supervisor Dr. Mousa Tawfiq AL-Akhras for his continuous encouragement, guidance, and support during the preparation of the thesis, and deep gratitude to my co-supervisor Dr. Azzam A. Sleit.

I also would like to express my appreciation and respect to all professors and doctors of the department of Computer Science for their great effort that they did during my studying at IT College.

LIST OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Meaning
AI	Artificial Intelligence
ANN	Artificial Neural Network
BGP	Border Gateway Protocol
BPNN	Back Propagation Neural Network
CCP	Centralized Control Protocol
DCP	Distributed Control Protocol
DEJAVU	Decompose Evaluate Join Append Verify and Unplug
DNS	Domain Name System
GA	Genetic Algorithm
LR	Learning Rate
MDS	Maximum Download Speed
MC	Momentum Coefficient
MS	Mirroring Server
MSE	Mean Squares Error

BEST UTILIZATION OF MIRRORING SERVER RESOURCES USING ARTIFICIAL NEURAL NETWORKS

By

Radwan Mohamed Radwan Al-Shalalfa

Supervisor
Dr. Mousa Tawfiq AL-Akhras

Co- Supervisor
Dr. Azzam Sleit

ABSTRACT

Due to the fact that huge amount of data are available to any user on the Internet and this amount increases with time, this implies huge number of requests from the Internet users to servers to download or process their data. Consequently, the need to increase the reliability and performance of such servers become an importance subject to tackle.

Replicating servers is a way of reducing overhead of the Internet users request as well as increasing reliability and performance of servers, but there is also a need to redirect user requests to one server from those replications so those servers will be unknown by the Internet users, this technique is called mirroring servers. Many researchers have been carried out to investigate models or techniques to enable selecting a server for any given request from the Internet users to the mirroring servers.

In this research, one of the Artificial Intelligence (AI) systems called Artificial Neural Networks (ANN) is used to select the appropriate server (best server) for new user's requests; our ANN models takes current servers features along with the new user request as input and give the selected server as output. The ANN needs to be trained to determine the best values of its weights. In order to do this training, we used results from other techniques as a ground truth, and such results are obtained by manual (human) selection.

We also compared the effectiveness of our neural networks with two different techniques human selection after eliminating the time needed from the human to make his/her selection, and

round robin selection. This comparison shows that the proposed model takes the advantages of these two techniques which are the speed of selection for the Round Robin selection methods and selecting the best server based on mirroring server features from the Manual selection method and overcome their disadvantages which are the selection method from the Round Robin and the need of human intervention for the manual selection. The Proposed model makes the selection decision taking the manual selection as a ground truth.

The ANN improves the utilization of the Mirroring Servers by 10% better than the Round Robin selection method, since in Round Robin selection method in more than 25% of the total time most of servers are idle and do not have any more requests to serve.

Chapter 1

Introduction

Server Mirroring techniques have been used for many years to increase the reliability and the performance of servers that receive heavy requests from too many users over the Internet.

In this chapter we will discuss the motivation that inspired us to perform this research on this subject and main problems that need to be solved to achieve the desired performance. This chapter is divided into fourth sections; the first section we will describe the importance of mirroring server, in the second section we will discusses the problem of the mirroring server, and in the third section we will discusses the main contribution of this thesis, and in the final section we will mention the organization of the remaining chapters.

1.1. Research Motivation

The use of the Internet is growing rapidly; this growth raises the need to split the load of the users to several servers called mirroring servers. Mirroring server's technology does such load balancing. This newly introduced technology solves other problems other than load balancing problem. Some of those problems addressed by the mirroring server are as following:

1. Mirroring Servers are used for load balancing by splitting the requested job(s) to different servers to reduce the overhead of these jobs on the network since the servers is distributed at different networks.
2. Reducing the time needed to execute the requested jobs by using the best available resource (server) for that job.

3. Mirroring server will provide a fault tolerant system by redirecting jobs to functional, operational server when some servers go down. Thus high data availability is achieved.
4. In mirroring server no need from the client to decide when to switch from an overloaded server to a lighter server as the selection process should be automated.
5. Concurrent downloads can be employed to increase download speed by dividing the request over many servers

Since mirroring server technology introduces several problems, this motivated researcher to increase the efficiency and the reliability of such technology, so several models and techniques have been proposed to the problem introduced from this technique which will be discussed in details in the next chapter.

1.2. Problem Formulation

Since we have more than one copy (mirror) of the server, then there is a need to select one server from them to serve new clients' requests. This selection of the server will naturally affect the overall performance of the mirroring server as well as the process of serving new requests.

When the selection of the server does not taken carefully into consideration of the current stage of the mirroring server then a mirroring server will reach a stage where it cannot serve any more request (which mean it becomes overloaded) and may be one of the server will goes down.

Therefore, in order to have correct server selection decision for any new request, historical background about the mirroring servers' behavior is needed as well as studying the current servers' status. This makes the selection of the server more accurate and it provides longer life time for the overall system.

1.3. Contribution

In this thesis we will use Artificial Neuron Networks (ANN) to find a relation or an association pattern between the input data and the output. The input data could be some of the available server features or the network bandwidth and the output will be the selection of the best server among them that best suited for a given request.

Since ANN is one of the Machine Learning (ML) techniques, this makes it one of the available solutions to learn from the available training data. One of the strength points for the ANN is its ability to find hidden relations between the input data and desired output as well as memorizing those training data as we will discuss them in the coming chapters.

The contribution in this thesis does not only depend on the ANN, we also need to select the feature's that will make effect on the server which is going to be selected, and such selection will affect the overall system performance and reliability

1.4. Organization of the Thesis

The reset of thesis is organized in four chapters as following. In chapter 2 describes the reasons behind introducing the mirroring server, the challenges raised after introducing such technique, and how previous researchers managed to solve the selection problem. In chapter 3, the proposed solution to address the selection problem is illustrated and how the problem is modeled. In chapter 4, discusses the results of experiment executed under the simulation networks and comparing the proposed techniques with two different techniques. In chapter 5, describes our conclusions from our research along with proposals for future work.

Chapter 2

Literature Review

The use of the Internet is growing rapidly over the last few years, as a result from this the network accesses from users tend to be heavily concentrated on some major servers which provide valuable data or processes to most Internet users, and the network around these servers is also congested. To distribute the load over multi-servers, a mirror server technology has been widely used.

This chapter describes the reason behind introducing this mirroring server technology, the challenges raised after introducing this technique, the performance and reliability of this technique on the client and server, how previous researchers tried to solve those challenges of the mirroring server techniques and finally some knowledge about artificial neuron networks structure and properties.

2.1. Server Mirroring

Every mega organization has E-processing and using the IT on huge demand to share important data, such data should be available twenty four hours seven days a week for all customers and employees. To archive this, multiple copies of the servers have been placed in several locations, and in order to manage these servers, mirroring techniques had been applied.

Mirror servers have been employed for many years on the Internet as a way to increase the reliability and the performance of processing client's requests in the presence of frequent access by many clients (Myers et al., 1999).

While mirroring can provide much higher aggregate throughput to a given data item, individual clients must choose a mirror server carefully to achieve reasonable performance.

Unfortunately, only ad hoc mechanisms for choosing the appropriate mirror server are currently employed (Myers et al., 1999).

Therefore, there is a need for an intelligent system that will study the current mirroring system features and make the best choice of the mirroring servers to maintain the current system performance and reliability with the increase in the number of concurrent users that can be served at the same time.

Upon a client request, a major challenge of choosing the best server among the available servers is the existence of many features that can be used as criteria to decide which server to use for this client request, and the speed of such selection should be fast enough to avoid delay in the client request.

Among the possible features there are the following features; the available memory for each server, available bandwidth, and average response time. The effect of these features can be changed during the life time of the mirror servers, that is why many different algorithms and techniques are exist which attempt to solve the selection problem.

To improve the performance and reliability of such mirror servers, many researchers have been working to study the current mirroring system features and their effect on the total performance. Such solutions have used many different optimization algorithms such as Brutal Force, Genetic Algorithm (GA) and Round-Robin to select the best mirroring server.

Cardellini et al. (1999) looked in updating web cluster architecture in which the Domain Name System (DNS) server, dispatches the user requests among the servers through the URL-name to IP-address mapping mechanism, is integrated with a redirection request mechanism based on HTTP protocol. The authors also compared the use of different mechanisms, and they

concluded that combination of centralized and distributed dispatching policies achieve the highest load balancing.

Rodriguez et al. (2000) proposed a scheme in which clients access multiple mirror sites in parallel to speedup document downloads while eliminating the problem of server selection, using this dynamic parallel-access lead to dramatic speedups in downloading documents, and the load is shared among servers without the need for a server selection mechanism.

Jamin et al. (2001) looked whether the current placement of the mirroring server over the Internet affects the overall system performance using some of the mirror placement algorithms and heuristics algorithm. The authors first used an algorithm called Min k-center which finds a set of center nodes (mirror servers) to minimize the maximum distance between a node and its closest center. The second used algorithm is Transit Node which searches in the out-degree which is defined as the number of other nodes that are connected to the node.

They model the Internet as a Graph, G= (V, E), where V represents set of nodes. There nodes could be candidate host where mirroring can be placed (H), set of mirrors for particular server (M), and clients (B) and E represent set of links between nodes (sub-set from V X V) the graph is weighted by the cost of the shortest path between two connected node. All edges are sorted in non-decreasing order. From the model, node H can only host mirror thus they run min k-center on the G with $V = H \cup B$ and $H \cap B = \varnothing$.

Gautam (2002) attempted to determine the optimal number and locations of proxy servers in a network to minimize throughput delay and demand constraints, by assuming that client or user requests at any location will always be sent to the nearest server this allows modeling each client–server as an independent queuing network. In order to solve this problem the author used a heuristic algorithm called Decompose Evaluate Join Append Verify and Unplug (DEJAVU) which was reported to be superior to the use of GAs.

Matthur and Mundur (2003) attempted to avoid participation of the client in balancing the load across servers, to achieve this; they proposed two protocols, Centralized Control Protocol (CCP) and Distributed Control Protocol (DCP). In CCP protocol**,** servers periodically send state information to the central server, indicating their current load. In DCP protocol a set of servers form a token passing arrangement to serve each client's request.

McManus (2005) used a heuristic algorithm to select the closest available web server from a group of mirrors; the heuristic is based on Border Gateway Protocol (BGP) as path lengths and can be determined without the introduction of any additional measurement traffic into the network.

Cai and Chow (2007) attempted to enhance the download of file in the mirror servers by representing two problems that firstly finding the maximum parallel download speed without restricting the number of mirror servers and secondly finding the best group of k servers for parallel downloads. The authors represented the problem as a graph and suggested two ways to solve these problems first by Brutal Force algorithms from which they got the worst case scenario O(n) and O(n^k) for the first and second problems, respectively, second way the authors used fixed length Genetic Algorithm and a variable-length Genetic Algorithm.

They proposed a Graphical Model, G= (V, E), where V represents the set of nodes these nodes could be the mirror servers, the clients, and routers between the clients and the mirror servers and E represents the set of edges or link segments that connect the nodes in the network. The weight for the Mirroring Server (M) which denoted as *speed (M)* is the document retrieval and transfer speed of M, and edge E which denoted as *speed (E)* is the available bandwidth on edge E. They used this graph to calculate the Maximum Download Speed (MDS).

Sleit et al. (2007) used GA to select a server among the available mirroring servers to distribute query in order to archive load balancing, the authors depend on two main features in the mirroring server: average server processing time and average reply time. They modeled the problem by a main server called load balancing server. The client request is transformed to the load balancing server which will redirect the request to the appropriate server using Genetic Algorithms (GA). The GA depends on two main attributes in such decision, average server processing time (Tp) which is the time request to process the client request and average reply time (Tr) which is the time to send the requested to the client.

Webb et al. (2007) let DM_n denote a delay matrix of size n x m for all clients in P= {P_1, P_2… P_n} and mirrors in M= {M_1, M_2… M_m}. An element is in i^{th} row and j^{th} column of DM_n matrix represents difference between the time a mirror M_j receives an update and the time the update was sent by client Pi. They assumed that the problem client-to-mirror-assignment (CMA) be a set of all client-to-mirror assignments for DM_n, and defend CMA_n to be optimal if $D = \Sigma\, d_{i,j} * x_{i,j}$ is minimized where $x_{i,j} \in \{0,1\}$ is equal to 1 (0) if P_i is (is not) assigned to M_j.

Nakaniwa et al. (2009) proposed a model named optimal mirror allocation; they presented the topology of the network as adjacency matrix A, whose elements have weighted by the distance between a pair of nodes. They calculated the shortest path matrix Q, and formulated the reliability, the cost, and the delay by the matrix Q.

2.2. Artificial Neural Network

Since the invention of the digital computers, humans have attempted to create machines which directly interact with the real world without their intervention. In this sense Artificial

Intelligence, in general and the Artificial Neural Networks (ANN's) in particular represents an alternative for endowing the computer one of the characteristics that makes the difference between humans and other alive being, the intelligence.

Artificial neural networks (ANN) have been developed as generalizations of mathematical models of biological nervous systems. A first wave of interest in neural networks (also known as connectionist models or parallel distributed processing) emerged after the introduction of simplified neurons by McCulloch and Pitts (1943), (Abraham, 2005).

The basic processing elements of neural networks are called artificial neurons, or simply neurons or nodes. In a simplified mathematical model of the neuron, the effects of the synapses are represented by connection weights that model the effect of the associated input signals, and the nonlinear characteristic exhibited by neurons is represented by a transfer function. The neuron impulse is then computed as the weighted sum of the input signals, transformed by the transfer function. The learning capability of an artificial neuron is achieved by adjusting the weights of the connection in accordance to the chosen learning algorithm, (Abraham, 2005).

Figure 1 illustrates the actual activity of the neural cell in any artificial neural networks, where connections (synapses) w_{ki} transfer the input signal (stimulus) x_i into the neuron cell, all transferred signal (stimulus) are summed inside the neuron cell after multiplying them with the corresponding connection (synapses) weight ($V_k = \sum_{j=1}^{p} W_{kj} * X_j$). Finally an activation function controls the amplitude of the output, for example an accepted range of the output is usually 0 and 1 or it could be -1 and 1.

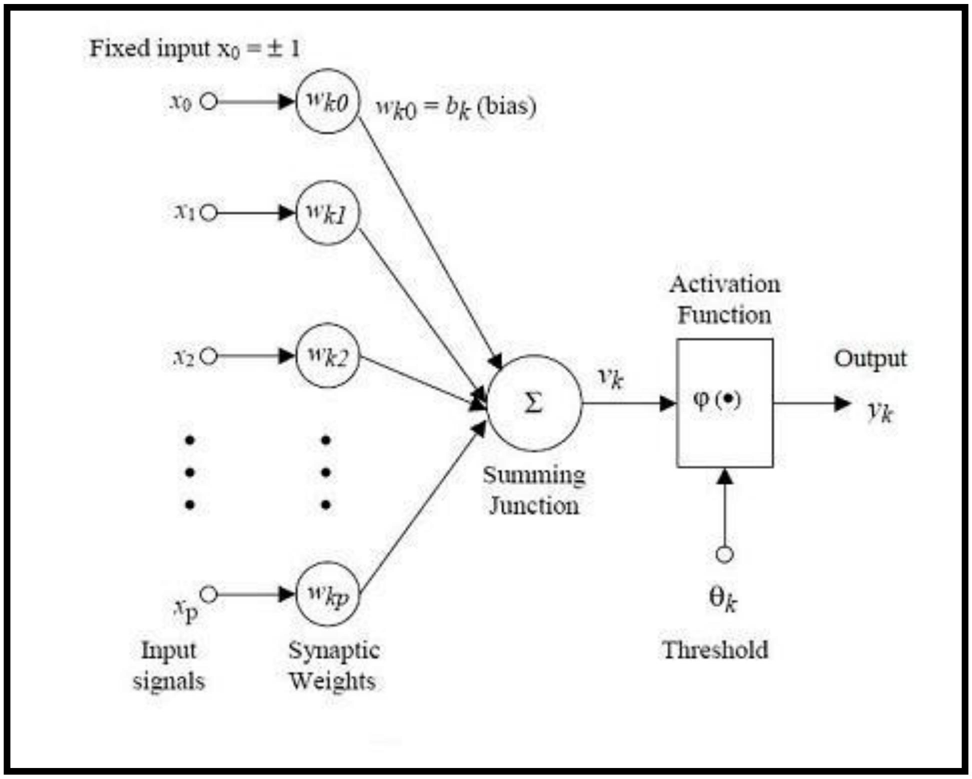


Figure 1: Structure of single neuron in the ANN (Bajpai et al., 2011)

Some researchers are still studying the neurophysiology of the human brain, but much attention is now being paid to the general properties of neural computation, using simplified neural models, these properties include (Joe, 1995):

1. **Trainability:** ANN can form matches between any input data and output pattern. This can be used for example in the mirroring selection problem where we need to find a connection between the input servers' features and the selected server.
2. **Generalization:** ANN does not just memorize the training data rather than it learns the underlying pattern between the input and the output, so it can learn new data from the training data, this also essential in the mirroring servers selection problem where

new status for each server can be found which is not mention at any item in the training set.

3. **Nonlinearity:** ANN can compute nonlinear, nonparametric functions of their input, enabling them to perform arbitrarily complex transformations of data. This is useful since the selection problem is a highly nonlinear process
4. **Robustness:** ANN is tolerant to both physical damage and noisy data; in fact noisy data can help the networks to form better generalizations.
5. **Uniformity**: Networks offer a uniform computational paradigm, which can easily integrate constraints from different types of inputs.
6. **Parallelism**: Networks are highly parallel in nature, so they are well-suited to be implemented on massively parallel computers, this will ultimately permit very fast processing of selecting the best server.

ANN includes a number of nodes and layers as following (Kaastra, 1996):

1. Number of input nodes: These nodes correspond to the input features.
2. Number of hidden layers: These hidden layers provide the ANN with the ability to generalize.
3. Number hidden nodes: There is no magical formula for selecting the best number of hidden neurons, although there are a number of rules of thumb that are available for calculating the number of hidden neurons e.g. number of hidden nodes = (number of input + number of output) * (2/ 3).
4. Number of output nodes: ANN with multiple outputs is especially widely used, will produce inferior result when comparing to network with single node.

There are several types of architecture of ANN, most type in use are:

1. Feedforwared ANN: The signal is only travel from input to output and it cannot go back (there is no feedback) they are used in pattern recognition problems.
2. Feedback/recursive ANN: The signals are travelled on both directions from input to output and from output to input, this type of network is dynamic and their status changed.

Learning methods of the Artificial Neural Network can be classified into three types:

1. Supervised learning, in this method we are concerned in finding the best function that associate the predefined input data with the output mainly it uses the Mean Squared Error to calculate the different between the actual result and the expected result, example of problem used this learning type is pattern recognition.
2. Unsupervised learning, in this method we concern in finding the best function that generate the minimal associated between the input data, example of problem used this learning type is clustering.
3. Reinforced learning, this aim is to discover a policy for selecting actions that minimizes some measure of a long-term cost, example of problem used this learning type is games.

Back-propagation algorithm (supervise learning) is one of the training methods that employs a gradient descent technique to adopt the neural network weights to minimize the mean squared error difference between the neural network output and the desired output, we denote the connection weight between neuron n_i and neuron n_j with W_{ij} therefore these weights can be represented using the adjacency matrix, so this algorithm works as following (Heaton Research, 2011):

1. Initialized the neural network training weights randomly

2. Calculate the sum up for each neurons(n_i) that are connected to the hidden neuron n_j using the following equation

$$y_j = f(\sum_i x_i W_{ij})$$

Where x_i is the result for the i^{th} neuron in the previous layer (or input data if there is not a previous layer), W_{ij} denotes the connection weight between neuron n_i and neuron n_j and f is the activation function. One of the activation function is the sigmoid which have the following equation

$$y_j = (1 + e^{-x_j})^{-1}$$

3. Compute the mean square error of one output neuron over all n, examples is the difference between the target t and actual a network output

$$mse = \frac{1}{n}\sum_i^n (t(i) - a(i))^2$$

Where $t(i)$ is the target output for neuron n_i and $a(i)$ is the actual network output for the neuron n_i

4. The change of the weight ($\Delta W_{ij}(k)$) between the neuron n_i and neuron n_j is as following

$$\Delta W_{ij}(k) = \eta\delta_j \mathrm{x_i} + \alpha\Delta \mathrm{W_{ji}}(\mathrm{k} - 1)$$

Where η is a parameter called the learning coefficient, α is the momentum coefficient, and δ_j is a factor depending on output neuron or a hidden neuron, for the output neurons it is defined as:

$$\delta_j = \frac{\partial f}{\partial net_j} (y_j - ynet_j)$$

Where $net_j = \sum_i x_i W_{ji} y_j ynet_j$

For the hidden neuron

$$\delta_j = \frac{\partial f}{\partial net_j} \sum_q W_{qj}\delta_q$$

The difference between the target and the actual neural output of a hidden neuron n_j is replaced by the weighted sum of $\delta_q$which terms already obtained for neurons n_q connected to the output of n_j.

Thus, iteratively, beginning with the output layer, the δ term is computed for neurons in all layers and weight updates determined for all connections according to mean square error equation above.

From the properties of the ANN such as trainability, generalization and parallelism, it is best suited for the Mirroring Server selection problem, in other words it is exactly matches the problem needs since it can be used to benefit from any algorithms used to solve the same problem but even with better and faster processing.

Chapter 3

Neural Load Balancer

The main goal of this research is to use machine learning for selecting the best server for a given request to the mirror servers. Other goals include using existing features to improve the accuracy of the neural network selector also comparing neural network selector with other selection methods. The results are evaluated using different performance metrics; system performance, execution time, server delay, and memory.

In our system the Internet servers and clients were simulated as points distributed in two dimensions area (XY-axis). In this 2D area clients and servers are replaced randomly inside the defined area, each client has direct access to the server, path from any client to each server has a lot of routers and is viewed by a single line between the client and each server. Any request from the client is passed first to the Main Server (decision server) which has a priori knowledge about each server's hardware specification such as CPU and Memory and status such as available Memory and CPU along with all assigned requests. This server is responsible for selecting the best server from the available servers; the selector performs the following major steps:

1. Evaluating the new client request.
2. Evaluating the current system status.
3. Depending on the above two steps select the best server in the available servers.
4. Assigning the selected server to the new request.

Figure 2 illustrates the process of training the ANN which begins with generating the training set for the ANN by examining different situation of all servers in the mirroring server in

different times and selecting the most appropriate server for each situation. These data are then passed to a newly created ANN to determine the best values for the hidden parameter and constants. Such data is generated from a selection method called manual selection. The back-propagation algorithm is used as a learning algorithm for the proposed neural network, this training ANN is used in the Main Server for selecting the server for any newly situation other than the one found in the training set.

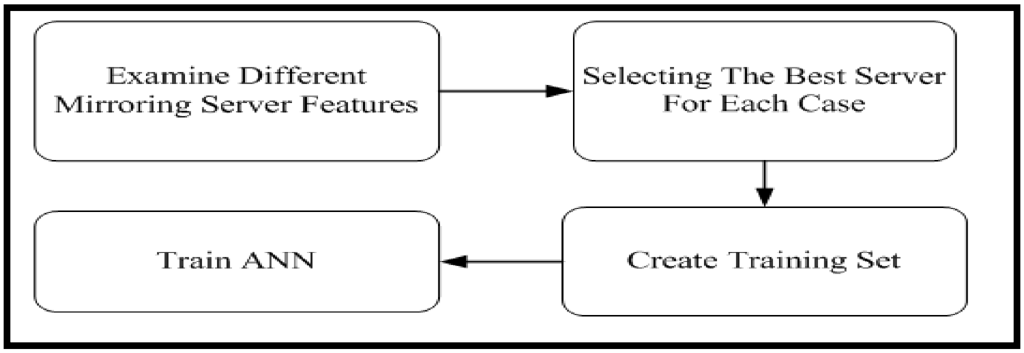


Figure 2: Process of training ANN

Figure 3 illustrates the processes that are carried out at the Main Server. First it receives the request from any client. All requests from clients are served one by one in queuing order. The server examines the current status of all servers of the mirroring environment upon serving the client request e.g. CPU used at each server, the physical memory available at each server, and the time expected to the server to finish the current running request. These values are then converted into format that the training ANN can accept, and also be passed to the network as input and the output of trained ANN is the server that most appropriate for the input client request, the client request is then forwarded to the selected server which is the output from the ANN .

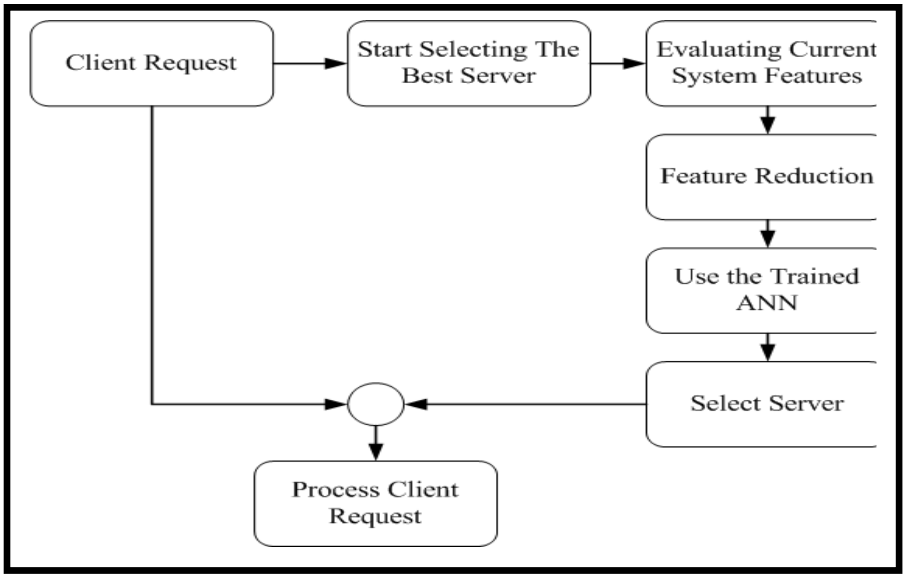


Figure 3: Main Server process

We divided the implementation of the proposed solution into two parts:

1. First part covered the processes of the Main Server; the pseudo code for this part is illustrated at Figure 4.
2. Second part covered the processes of other servers the pseudo code for this part is illustrated at Figure 5.

```
start simulation
while the simulation is not stopped
begin
  for each request from the client that are not assigned
  begin
    examine all status for each server and the need for the client request
    transfer the collected data to the trained ANN
    examine the output of the ANN then determine the   selected server
    pass the client request to the selected server
  end
  increment the simulation time
end
```

Figure 4: Main Server process algorithm

```
start simulation
while the simulation not stopped
begin
   for each server in the system
   begin
      determine the number of slots that the server can generate in single round
      serve request assigned to the server as much as the server slots
end
increment the simulation time
end
```

Figure 5: Servers processing algorithm

Feed-forward type for the proposed Artificial Neural Networks is being used. The used ANN consists of three layers as following:

1. First, the input layer which accepts the features of the mirroring servers as input, so it will contain neurons as much as the input features, and since we used the server CPU and memory along with the request slots and memory that means our ANN will consist from 12 neurons as we have five servers with two features each and two features for the requests which are the required slots and memory, the neurons in this layer do not perform any transformation of the input data or in another word it used the pure-line transfer function.
2. Second, the output layer which is responsible for selecting the best server and it consists of 2 neurons, because we have 4 servers and used the binary presentation for server’s id. Only 2 neurons will be needed at this layer. This layer used the logsig transfer function which produced value between 0 or 1 as well as meeting our requirement for binary presentation.
3. Finally the hidden layer which we used a rule of thumb rules that used to determine the number of hidden neurons which is

$$\text{number of hidden nodes} = (\text{number of input} + \text{number of output}) * (2/3)$$

So 9 neurons is going to be used in this layer, tansig transfer function is used as well by layer neurons.

We used the trainlm as training function for our artificial neural network, since trainlm is the fastest back-propagation algorithm and the most recommended for supervised learning, Appendix B mention the function that is used as ANN.

The simulated networks have been built which contains a number of servers and brokers (Main Server) which are responsible for delivering the request to the appropriate server that are spread into a predefined area and number of clients which are also spread into the same area.

In order to compare these techniques, we have built networks contains predefined structure of servers and clients but with different requests. This network is connected with the following main components:

1. **Servers**: Five servers has been generated to serve the network clients, each of them have the following hardware specifications:
   a. Memory: Each server has 2–4 Gaga Byte (GB) of memory.
   b. Central Processing Unit (CPU): Each server has 2-5 Gaga Hertz (GHz) of CPU.

2. **Clients**: 200 clients have been generated to request service from the above defined servers.

3. **Brokers**: One of the servers defined above is used as a broker.

Chapter 4

Experiments

Three different selection techniques have been studied in the simulated networks in order to have clear decision about the performance of the proposed technique. Appendix A describes the simulation program. The three selection techniques are described in section 4.1, section 4.2 are also discussed the results, and in section 4.3 discuss the summary of results.

4.1. Selection Methods

1. **Round Robin**: is an arrangement of choosing all elements in a group equally in some rational order, usually from the top to the bottom of a list and then starting again at the top of the list and so on(Cao, 2004). This technique is used to implement one of the chosen techniques by selecting the first available server (element) and then the second (element) till the last server and then start again from the first available server (element) and so on. This technique usually does not take any of the server hardware specification or current status into account in its decision, which makes this technique not the best selection technique for the mirroring server.

2. **Manual**: is an arrangement of selecting all elements in group, according to some criteria changed from one time to another depending on user interaction. This technique is also implemented as one of the choosing techniques. Two different criteria are used to select the best server (element):

    a. CPU time: selecting the best server depending on the time taken from the server to finish the current assigned request.

b. CPU time and request time: selecting the best server depending on the time taken from the server to finish the current assigned request and also this new request.

3. **Neural Load Balancer**: ANN is trained depending on the results of the second technique (Manual) as a ground truth.

The simulated networks used the above techniques to select the server which is the best among the other servers for randomly generated requests from different clients. At each time during the live time of the simulation networks, these requests have been generated as follows:

1. First experiment: executed within 10 seconds from the simulation time and serve 200 randomly generated requests which have been started at different times during the simulation.
2. Second experiment: executed within 20 seconds from the simulation time and serve 500 randomly generated requests which have been started at different times during the simulation.
3. Third experiment: executed within 30 seconds from the simulation time and serve 1000 randomly generated requests which have been started at different times during the simulation.
4. Fourth experiment: executed within 40 seconds from the simulation time and serve 10000 randomly generated requests which have been started at different times during the simulation.
5. Fifth experiment: executed within 50 seconds from the simulation time and service 100000 randomly generated requests which have been started at different time during the simulation.

The first three experiments have been executed on all three selection method and the reaming experiment have been executed on round robin and neural load balancer since it takes time from the user in the manual selection to select the best server.

All of the above experiments have been generated randomly more than one time but the results from executing these experiments have approximately the same result.

4.2. Results

When running the above mentioned experiments, and in order to compare between the results of each technique, three measurements have been taken into consideration, these measurements are:

1. Waiting Time: This will measure the amount of slots that have been taken from any assigned request to be served at any server including the idle slots (waiting to another request to finish after it has started execution). The calculation of these slots starts after the request being served by the server; this will measure how much the server is overloaded. In the overloaded server the request will spend much of its time in idle status not executed at all which will cause delay in the request and overhead at the server side.
2. Memory: This will measure the amount of available memory at the server when attempting to assign a new request to the server.
3. Turnaround Time: which will measure the amount of time the request spent at the assigned server after it starts execution, each server has different CPU power which means that it can serve different number of requests at the same time, and this means that each server can provide different number of slots.

From the definition of each of the used techniques, we can produce some of the following behavior.

1. Since the Round Robin select the servers in the same order it will most likely split the requests equally between the servers, if we assume that each server will be available at any round, this will cause the same server delay since any request at any server will wait the same number of requests.

2. Since the Manual a select server according to the user criteria, it will reflect this criteria in the request result for example if we use the CPU as the criteria for selecting the best server then we will see the requests are split into different groups with different number of requests at each group according to each server's power and the most powerful server has the largest group. Since this method is affected by the user judgment then it will be the most powerful technique after eliminating it is main drawback which is the delay of selection process, due to the fact that the user is slower than the computer, and this causes overhead since it depends fully on the user and makes this selection method not applicable in the real work.

3. Since the ANN is used to find association between the input data and the output then it most likely will reflect the behavior of the data being used in the training stage if it can learn from such data, while Manual data is being used for training our ANN, then this ANN will most likely be functional similar to the Manual technique, which means that it will take the benefit of this method and eliminate the main problem which is caused by the delay of selecting the best server, by eliminating the user intervention.

All of the below figure shows the result from the first three experiment since they executed on all of three experiments.

4.2.1. Waiting Time

Since many requests will be assigned to each server, we have calculate the average of waiting time for each request and consider this value as the waiting time for the corresponding server itself at each experiment.

Figure 6 shows the waiting time at each server in each experiment executed at our simulation network which used the Round Robin technique as the selecting mechanism for the best server.

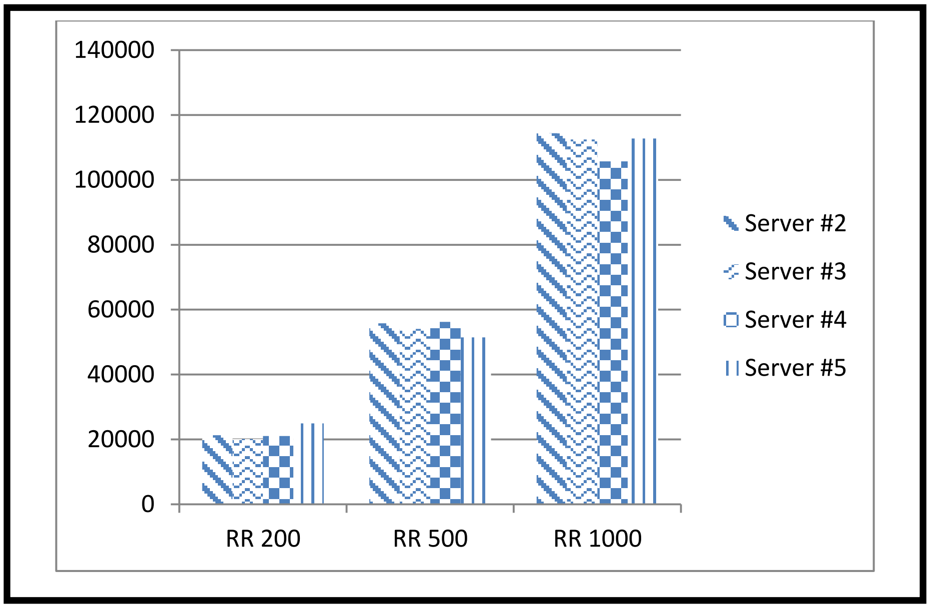


Figure 6: Waiting Time at each server after using the Round Robin

From Figure 6, it can be noticed the following points regarding the waiting time at each server when using the round robin selection technique:

a. All servers have the same or approximately the same as average waiting time, since the servers have the same number of requests that means any request at any server will wait the same number of slots regarding the power of the server, from this it can be concluded that the requests have the same number of delay in terms of slots.

b. The difference between the servers does not reflect the CPU power of the server, since each request needs different number of slots this means that request with the least needed slots will finish first and reduce the total number of assigned requests and that makes difference between the servers in the number of remaining requests in each one.

c. Since Round Robin technique finds the most available server in line, this means that the most powerful server will have more requests than the least powerful server, which will also cause the difference between the servers in terms of waiting time.

Figure 7 shows the waiting time at each server in each experiment executed at our simulation network which used the Manual technique as the selecting mechanism for the best server.

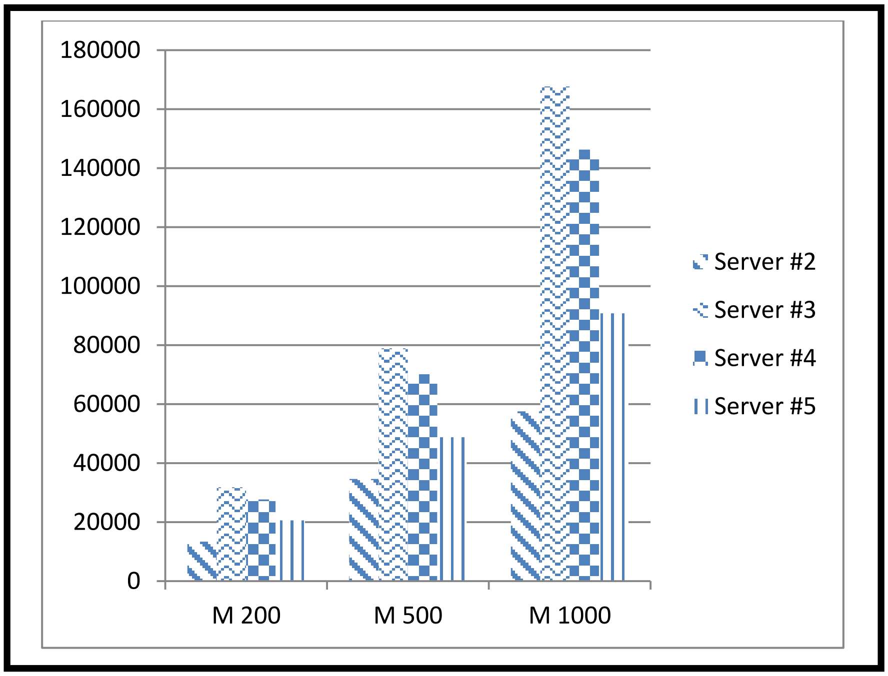


Figure 7: Waiting Time at each server after using the Manual

From Figure 7, we observe the following point regarding the waiting time at each server when using the manual selection technique. Each server has different average waiting time, since each server has different number of requests; which is caused by user selection. This means that any request at each server will wait different number of requests to be served before the cycle comes to it, which will cause the difference in the waiting time, most powerful servers have the most average waiting time and the least powerful server has the least average waiting time.

Figure 8 shows the waiting time at each server in each experiment executed at our simulated networks which used the trained ANN technique as the selecting mechanism for the best server.

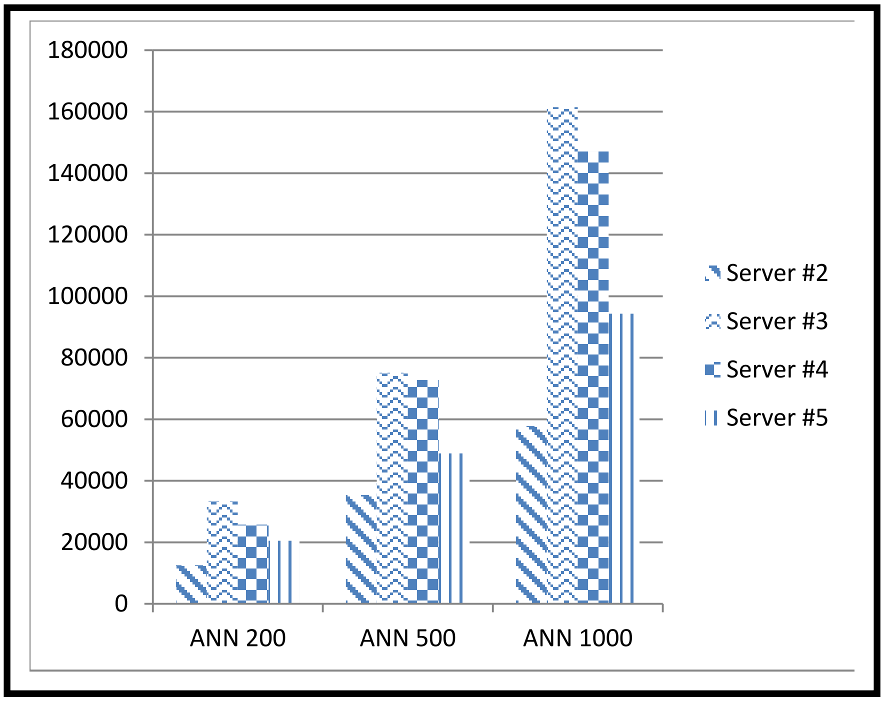


Figure 8: Waiting Time at each server after using the ANN

From Figure 8, we observe the following points regarding the waiting time at each server when using our ANN selection technique

a. All servers have different number of average waiting time, since the ANN network is used. ANN attempts to have the same behavior as the training set, that we used the manual selection technique above to do its training. The behavior of this trained ANN has approximately the same behavior as the manual selection method, which means different number of requests to be assigned at each server.
b. These differences in average waiting time are not the same as the Manual technique since this data has not been seen by the trained ANN.

4.2.2. Turnaround Time

Since many requests have been assigned to each server, we have calculated the average turnaround time for each request and consider this value as turnaround time for the corresponding server itself at each experiment.

Figure 9 shows the turnaround time at each server in each experiment executed at our simulation network which used the Round Robin technique as the selecting mechanism for the best server.

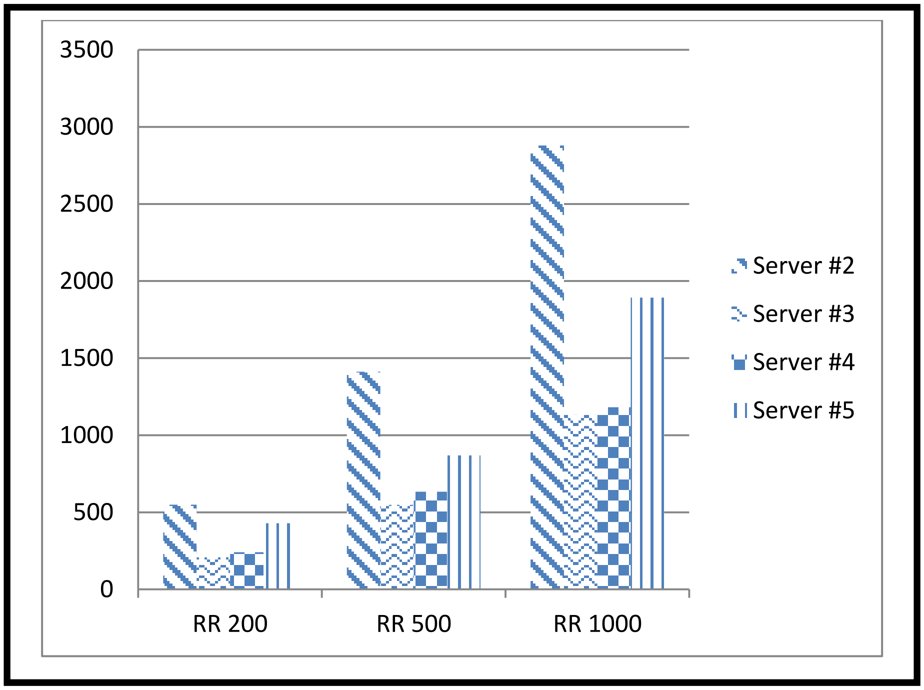


Figure 9: Turnaround Time at each server after using the Round Robin

From Figure 9, we observe the following points regarding the turnaround time at each server when using the Round Robin selection technique:

a. All servers have different average turnaround time, since all servers have the same number of requests, which means when the server is most powerful then it

will finish the assigned requests first and the least powerful server will finish the assigned requests last.

b. Average turnaround time observations do not contradict with the waiting time observations, since in each server different number of slots can be generated at any time depending on its CPU power, which means if a server can generate 5 slots at a time another server will generate 10 slots at the same time, which means the first server needs two times the number of slots as the first one (which is turnaround time).

c. The most powerful server has the least turnaround time, and the least powerful server has the most turnaround time.

Figure 10 shows the turnaround time at each server in each experiment executed at our simulation network which used the Manual technique as the selecting mechanism for the best server.

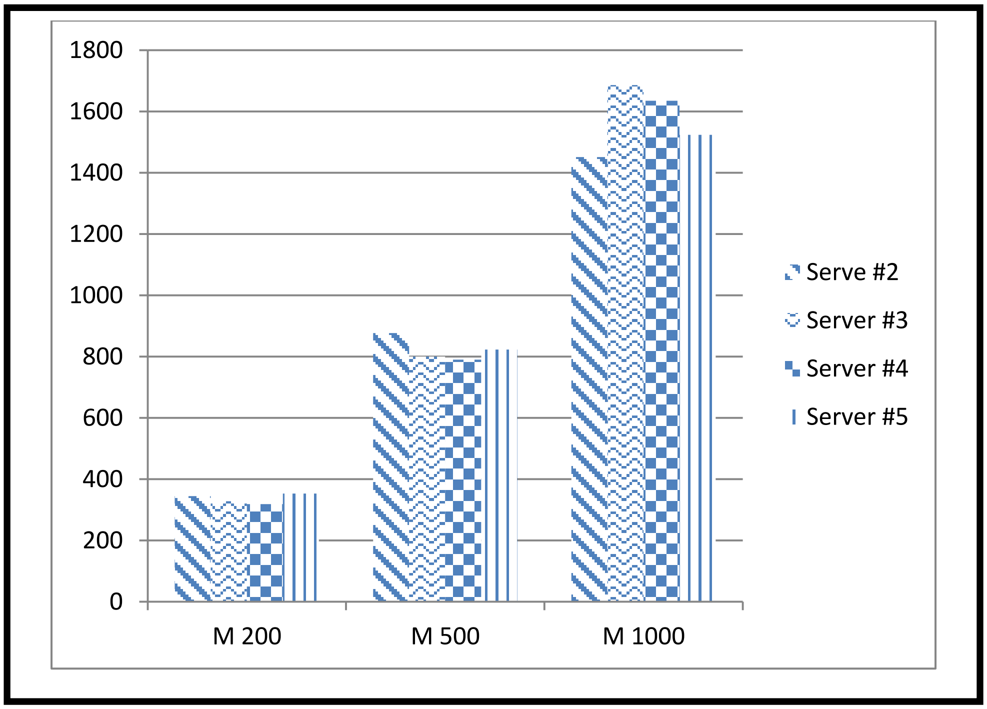


Figure 10: Turnaround Time at each server after using the Manual

From Figure 10, we observe the following points regarding the turnaround time at each server when using the Manual selection technique:

a. Each server has the same or approximately the same average turnaround time, since each server has different number of requests - caused by the user selection- this means that in order to the server to finish its request it depends on the user criteria for selecting server, which will lead that the most powerful server to have the most number of requests and the least powerful server to have the least number of requests, and that is why the servers will finish all requests approximately at the same time.
b. The difference between the turnaround time also reflects the power of the server, since each request needs different number of slots that will lead that the most powerful server will finish the request sooner than the least powerful server.

Figure 11, shows the turnaround time at each server in each experiment executed at our simulation network which used the trained ANN technique as the selecting mechanism for the best server. From Figure 11, we observe the following points regarding the turnaround time at each server when using our ANN selection technique:

a. All servers have same or approximately the same average turnaround time, since we used the ANN which attempts to have the same behavior as the training data which we use the manual technique above to do such training. That means that the behavior of trained ANN will have approximately the same behavior as the manual method, which includes the different number of requests to be assigned to each server.

b. Those differences in average execution time are not the same as the Manual technique since this data has not been seen by the trained ANN.

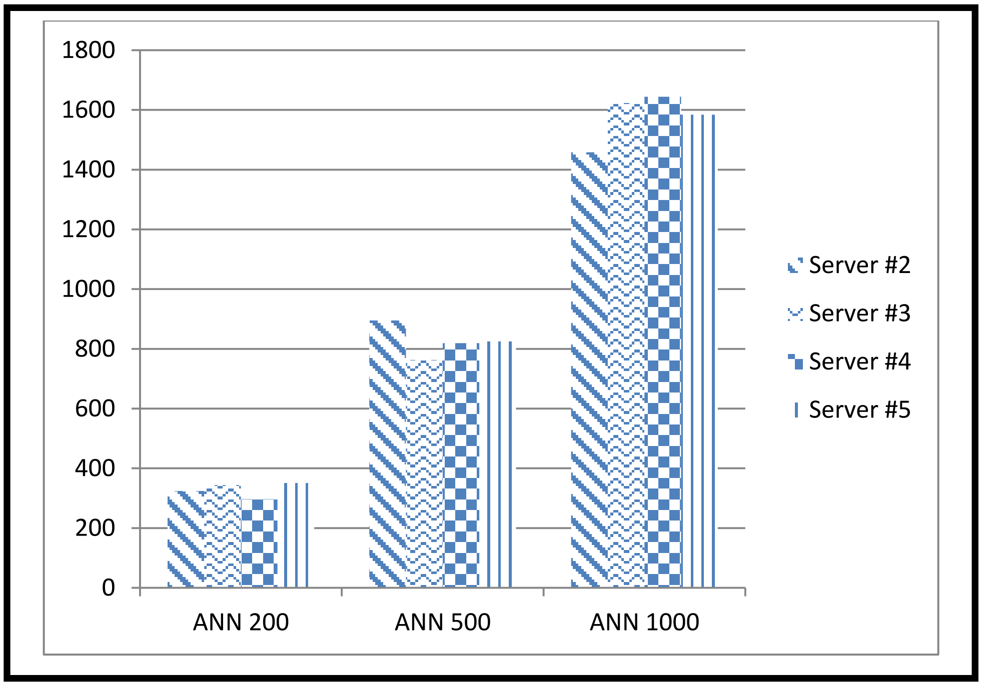


Figure 11: Turnaround Time at each server after using the ANN

4.2.3. Memory

Since many requests have been assigned to each server we have calculated the average of available memory at each server when assigning the new request to it, and consider this value as the available memory for the corresponding server itself at each experiment

Figure 12 shows the available memory at each server in each experiment executed at our simulation network which used the Round Robin technique as the selecting mechanism for the best server.

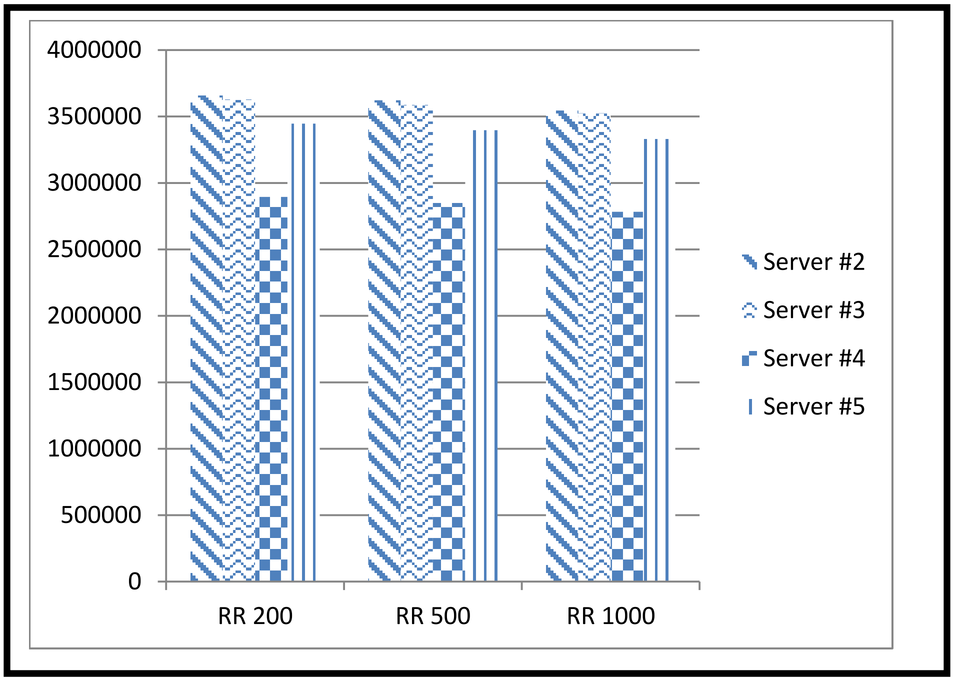
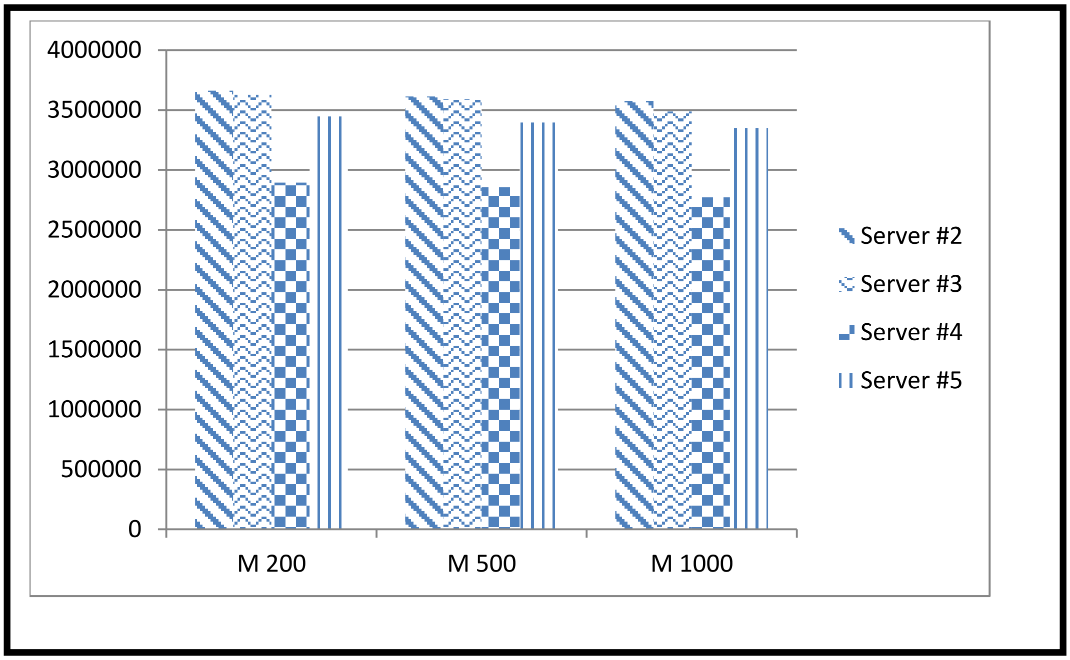


Figure 12: Available Memory at each server after using the Round Robin

From Figure 12, we observe the following point regarding the available memory at each server when using round robin selection technique. All servers have different available memory; since all servers have the same number of requests then the available memory at each server will depend on the total physical memory installed at each server. Server 2 has the largest physical memory and Server 3 has the smallest physical memory.

Figure 13 shows the available memory at each server in each experiment executed at our simulation network which used the Manual technique as the selecting mechanism for the best server.



From Figure 13, we observe the following point regarding the available memory at each server when using manual selection technique. All servers have different available memory, since all servers have the same number of requests then the available memory at each server will depend of the total physical memory installed at each server, but in the manual technique we can see that this technique attempts to minimize the difference between the servers' available memory.

Figure 14 shows the available memory at each server in each experiment executed at our simulation network which used the ANN technique as the selecting mechanism for the best server.

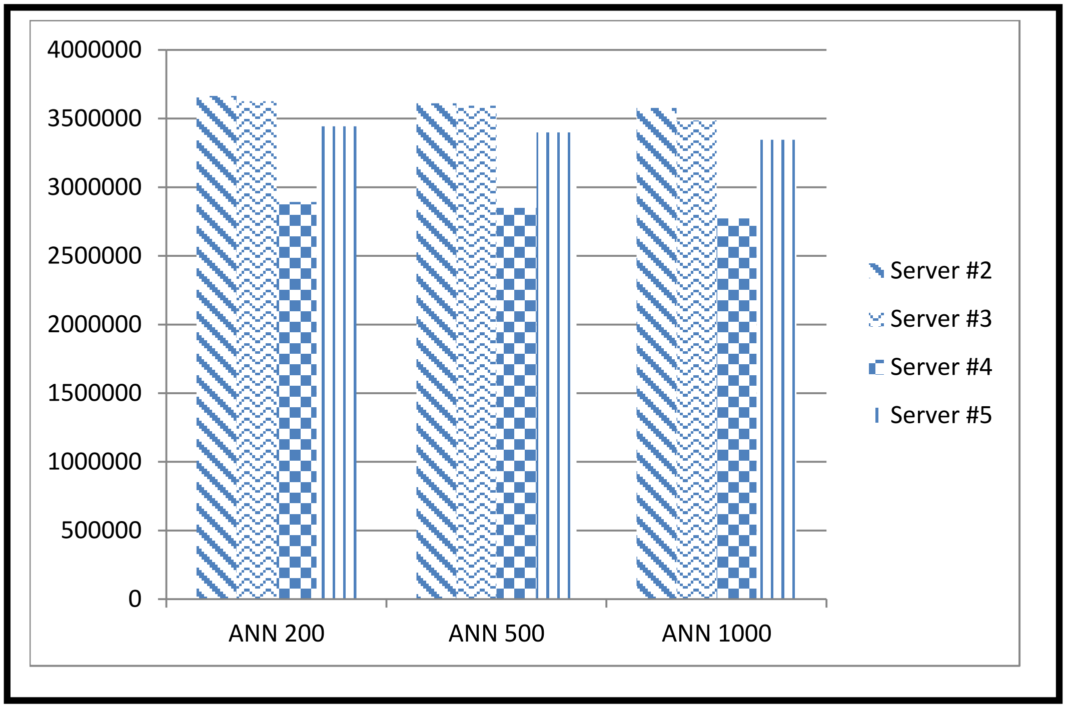


Figure 14: Available Memory at each server after using the ANN

From Figure 14, we observe the following point regarding the available memory at each server when using manual selection technique. All servers have different available memory, but we can see that the behavior of this technique is similar to the Manual technique.

4.3. Summary of Results

From all the above experiments the following comparison results between the three different selection methods is abstained.

First the Round Robin selection method makes the difference between the servers waiting times the minimum among all selection methods. This means all requests in all servers will have approximately the same waiting time, but in the Manual and ANN selection methods the difference between the servers waiting time is variant which means that the waiting time to all

request is different and it depends on the server hardware such as CPU and memory. These results are shown in the Table 1.

Server Number	Manual	Round Robin	ANN (Proposed model)
Second	57564	114392	57807
Third	167686	112475	161500
Fourth	146288	105692	147056
Fifth	90775	112757	94325

Table 1: Waiting Time for each server for the third experiment

Second, the Manual selection method makes the difference between the servers turnaround time the minimum among all selection methods which means that all servers will finish all assigned requests at the same time same as the ANN selection method but in the Round Robin method the servers with powerful hardware specification will finish all assigned requests first and the least powerful servers will finish all assigned requests last. Table 2 shows the turnaround time for all servers in the third experiment.

Server Number	Manual	Round Robin	ANN (Proposed model)
Second	1451	2879	1458
Third	1686	1132	1623
Fourth	1635	1182	1644
Fifth	1524	1892	1584

Table 2: Turnaround Time for each server in the third experiment

In general, the Manual selection method as well as the ANN selection method increases or decreases the waiting time for all requests to minimize the difference between the turnaround time between all servers.

Chapter 5

Conclusions and Future Work

4.1. Conclusions

In this thesis, we have studied the features of the servers that will affect our decision in selecting the best server in the mirroring server system. In this stage we reduce the feature that we are going to use in our proposed model (Artificial Neuron Networks), since if we reduce these features, the time of training stage for our proposed model (Artificial Neuron Networks) will be reduced as well.

Our Artificial Neuron Networks training data has been generated from another selection technique called manual selection which is the best selection techniques, after eliminating the delay of human selection, since the human can study the current system features and have correct decision using intelligence, but this method has a main drawback it is very slow because it needs human intervention for each new request, we used the ANN to solve this drawback and use the benefit of this selection technique to solve the mirroring server selection problem.

To obtain evaluation results from our ANN selector, we compared the ANN selector with two different techniques after running them on same environment specification (number of servers, number of clients and single broker) as well as same number of requests from each client, the result from running experiment shows that ANN selector produces approximately the same results generated from the manual selection method of course after overcoming its main drawback since the human intervention has been avoided in our ANN selector.

4.2. Future Work

As a future work we have planned to experiment different ANN topology and learning functions and compare it with the currently used topology as well as exploring more features of the mirroring server such as the distance and networks traffic and study their effect on the selection of the server along with testing our proposed idea on some of existing networks simulators such as Network Simulator-2(NS-2), Global Mobile Information System Simulator (GloMoSim).

We also planning to tackle the local minima problem of Back Propagation Neural Network (BPNN) training function by initialize the weights, and train the feed forward neural network using genetic algorithm.

References

Abdel-Hamid, Y.S. Gulliver, T.A. (2009), Improved Parallel Access to Multiple Internet Mirror Servers. **Micro-NanoMechatronics and Human Science, 2003 IEEE International Symposium on**, 1, 446-449.

Abraham, Ajith, (2005), Artificial Neural Networks. In: Peter Sydenham, Richard Thorn, **Handbook of Measuring System Design,** John Wiley & Sons, 901-908.

Bajpai, Saumya. Jain, Kreeti. Jain, Neeti(2011), **Artificial Neural Networks, International Journal of Soft Computing and Engineering (IJSCE)**, 1, 27 – 31.

Cai, Yu. Chow, C. Edward (2007), Algorithms for Selecting Multiple Mirror Sites for Parallel Download. **International Journal of Computer Science and Network Security**, 7(10), 273-278.

Cao, J.; Yang, L.T.; Guo, M.; Lau, F. (2004), **Parallel and Distributed Processing and Applications**, 3358, Springer.

Cardellini, Valeria. Colajanni, Michele, Philip S. Yu. (1999), Redirection Algorithms for Load Sharing in Distributed Web-server Systems. **Proc. IEEE 19th Int'l Conf. on Distributed Computing Systems (ICDCS'99)**, 528-535.

Gautam, N. (2002), Performance analysis and optimization of web proxy servers and mirror site. **European Journal of Operational Research**, 142(2), 396-418.

Heaton Research™ and Encog™ are trademarks of Heaton Research(2011), from: **http://www.heatonresearch.com/**

Joe, Tebelskis, (1995), **Speech Recognition using Neural Networks**. Unpublished Doctoral Dissertation, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213-3890.

Kaastra, Iebeling. Boyd, Milton(1996), Designing a neural network for forecasting financial and economic time series, **Neurocomputing**, 10(3), 215-236.

Liu, H. Jia, X. Li, D. Lee, C.H. (2004), Optimal placement of mirrored web servers in ring networks. **Communications, IEE Proceedings-**, 151(2), 170- 178.

Matthur, A. Mundur, P. (2003), Dynamic Load Balancing Across Mirrored Multimedia Servers, **2003 International Conference on Multimedia and Expo**, 2, 53-6.

McCulloch, W.S. and Pitts, W.H. (1943), A Logical Calculus of the Ideas Immanent in Nervous Activity**. Bulletin of Mathematical Biophysics**, 5, 115–133.

Myers, Andy, Dinda, Peter, Zhang, Hui (1999), Performance Characteristics of Mirror Servers on the Internet, **De Montfort University**, 1, 304-312.

Nakaniwa, A. Takahashi, J. Ebara, H. Okada.(2009), Reliability-Based Optimal Allocation of Mirror Servers for Internet, **Global Telecommunications Conference, 2000. GLOBECOM '00. IEEE**, 3(11), 1571-1577.

R. McManus, Patrick.(1999), A Passive System for Server Selection within Mirrored Resource Environments Using AS Path Length Heuristics, **Applied Theory Communications, Inc,** 1-10.

Rodriguez, Pablo, Kirpal , Andreas, W. Biersack, Ernst (200), Parallel-Access for Mirror Sites in the Internet", **INFOCOM 2000. Nineteenth Annual Joint Conference**, 2, 864 – 873.

Sleit, Azzam, Al-Mbaideen , Wesam, Alzabin, Nawal, Dawood , Hadeel, AL - Qarute, Khaled (2007), Efficient Query Processing Over Mirroring Servers Using Genetic Algorithms, **Neural Networks World**, 4, 311-320.

Sugih, Jamin, Cheng, Jin, Anthony, R. Kurc, Danny, Raz, Yuval ,Shavitt (2001), Constrained Mirror Placement on the Internet, **INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies**, 1, 31-40.

Webb , Steven D. Soh, Sieteng. Lau, William (2007), Enhanced Mirrored Servers for Network Games**, NetGames '07** , 117-122.

Webb , Steven D. Soh, Sieteng.(2008), Adaptive Client to Mirrored-Server Assignment for Massively Multiplayer Online Games, **Fifteenth Annual Multimedia Computing and Networking**, 68180I-1.

Yokota, H.; Kimura, S.; Ebihara, Y. (2004), A proposal of DNS-based adaptive load balancing method for mirror server systems and its implementation, **Advanced Information Networking and Applications, 2004. AINA 2004. 18th International Conference on**, 1, 208 – 213.

Appendix A: Simulation program

This appendix described our simulation program. The system is simulated as nodes scattered randomly into two dimension area (X, Y), in order to compare between different selection methods (1) Round Robin, (2) Manual and (3) Artificial Neural Network (ANN)

Our simulation consists of two parts (1) Processing which contains the main functionality of our simulation and (2) Storage which will store all information generated or used in the first part.

MATLAB programming language was chosen to implement the first part of the simulation (Processing) for the following reasons:

1. Since ANN is going to be studied as a selection method for the Mirroring Server (MS), so MATLAB is the perfect choice since it has ANN fully implemented.
2. More control on the simulation will be granted.
3. Networks effect will be neutralize, and this will give us more focused on the selection problem.

SQL Server engine will be used in the second part at storage to store the simulation data.

1. Database (Storage)

Database shown in Figure 15 contains four tables as following (1) Clients, (2) Servers, (3) Requests and (4) ServersHistory, each of them contains specific information about simulation objects.

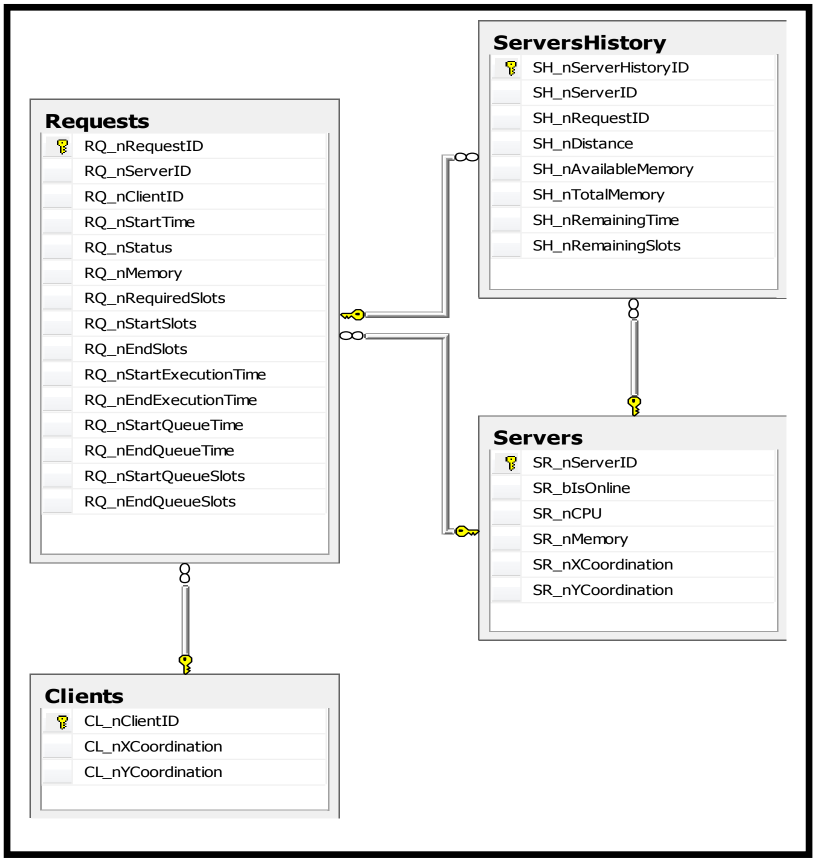


Figure 15: E-R DB diagram

Clients: Table that will store all the client position on the simulation area as pairs (x and y) and it contains the following columns:

1. **CL_nClientID**: Represents unique identifier to distinguish between clients in our simulation.

2. **CL_nXCoordination**: Represents the first part of client's position coordination in our simulation.
3. **CL_nYCoordination**: Represents the second part of client position coordination in our simulation.

Servers: Table contains all server information that will be used during the simulation live time and contains the following columns.

1. **SR_nServerID**: Represents unique identifier to distinguish between servers in our simulation.
2. **SR_bIsOnline**: Represents status of the server if it can serve client request or not.
3. **SR_nMemory**: Represents the amount of memory that is installed at the server.
4. **SR_nCPU**: Represents amount of CPU installed at the server (this value will be used to calculate the amount of slots that server can generate in a single simulation time).
5. **SR_nXCoordination**: Represents the first part of the server's position coordination in our simulation.
6. **SR_nYCoordination**: Represents the second part of server's position coordination in our simulation.

Requests: table contains list of requests that have been generated and processed during the simulation and contains the following columns:

1. **RQ_nRequestID**: Represents unique identifier to distinguish between requests in the simulation.

2. **RQ_nServerID**: Represents the server id (**SR_nServerID** above) which has been assigned for this request, any new request is assigned to the server id 1 (not exist in the tables because it have been consider as the Main Server).
3. **RQ_nClientID**: Represents the client id that generates this request.
4. **RQ_nStartTime**: Represents the time when this request was generated.
5. **RQ_nStatus**: Represents the status of the request (0) Creation, (1) In Progress, (2) Finished and (3) Queued.
6. **RQ_nMemory**: Represents the amount of memory that is needed by the request in order to be served at the server.
7. **RQ_nRequiredSlots**: Represents the number of slots the request needs from the server CPU to be finished.
8. **RQ_nStartSlots**: Represents the total number of slots that the server has been generated before it started processing this request
9. **RQ_nEndSlots:** Represents the total number of slots that the server has been generated before it finished processing of this request including the idle time (time when server is serving another request).
10. **RQ_nStartExecutionTime**: Represents the amount of time taken by the simulator to change the status of this request from order (0) to assign (1).
11. **RQ_nEndExecutionTime**: Represents the amount of time taken by the simulator to change the status of this request from assign (1) to finish (2).
12. **RQ_nStartQueueSlots**: Represents the total number of slots that the server has been generated before it queues this request
13. **RQ_nStartQueueSlots**: Represents the total number of slots that the server has been generated before it removes this request from the queue.

14. **RQ_nStartQueueTime**: Represents the amount of time taken by the simulator to change the status of this request from order (0) to queue (3).
15. **RQ_nEndQueueTime**: Represents the amount of time taken by the simulator to change the status of this request from order (3) to assign (1).

ServersHistory: Table contains the statues of each server when each request is going to be assigned and contains the following columns:

1. **SH_nServerStatusID**: Represents unique identifier to distinguish server status in the simulation.
2. **SH_nRequestID**: Represents the id of the request that is going to be assign when the status record is generated.
3. **SH_nServerID**: Represents the id of the server that has this status record.
4. **SH_nDistance**: Represents the distance between the client who created the request and the server (client of the **SH_nRequestID** and **SH_nServerID**).
5. **SH_nAvailableMemory**: Represents the remaining memory at the server after calculating all requests memory that already assigned to this server.
6. **SH_nTotalMemory**: Represents total memory installed at the server.
7. **SH_nRemainingSlots**: Represents the amount of the slots from the server to finish all request currently assigned to it.

2. Processing (MATLAB)

MATLAB program shown in Figure 16 contains the following operation (1) Connect, (2) Disconnect, (3) Area, (4) Servers, (5) Requests, (4) Prepare, (6) Star.

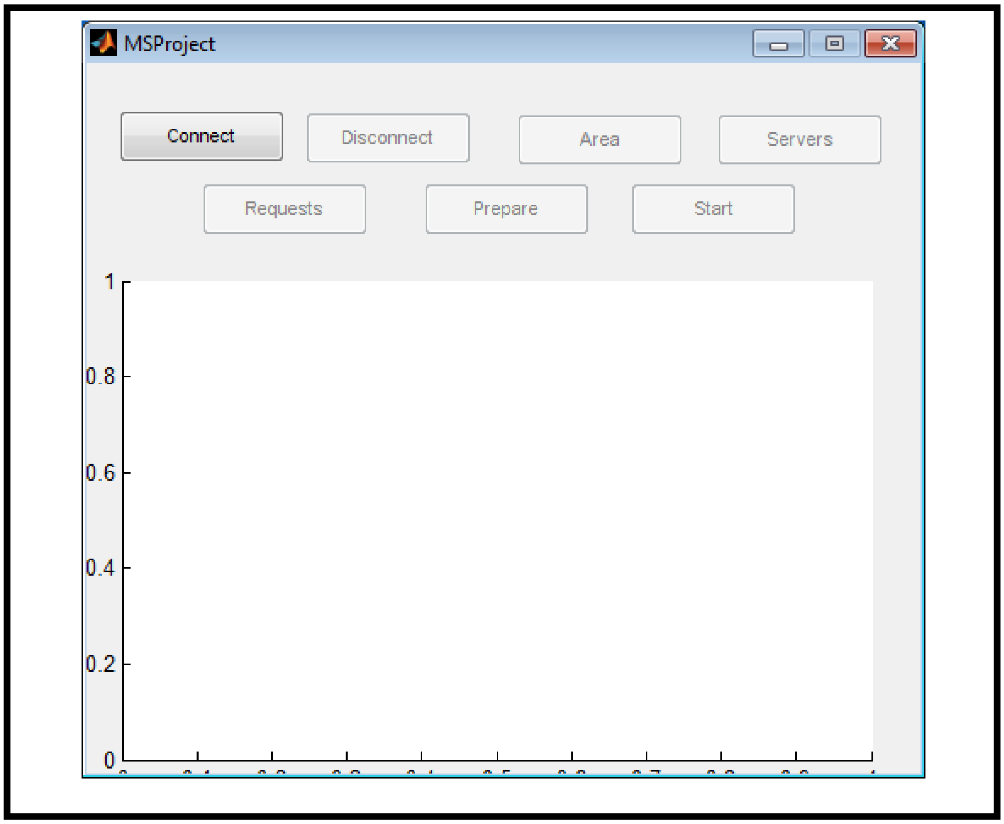


Figure 16: Mirroring Server simulator

Connect: This operation is used to connect simulation into the database, an input dialog asks the user to fill the proper connection information of the database store, and the dialog contains default connection values. Connect dialog is shown in Figure 17.

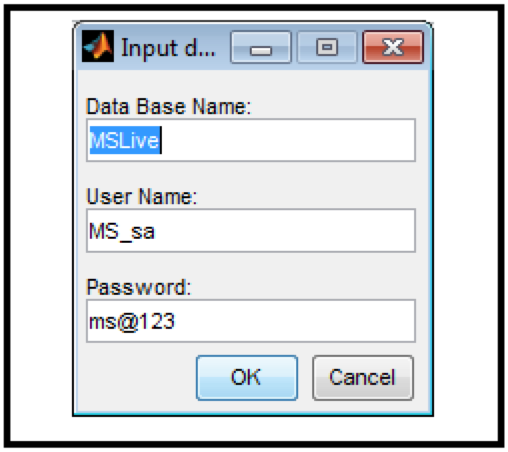


Figure 17: Connection information

Disconnect: the operation closed all connection between the simulation and the SQL server database.

Area: This operation set the area dimension in where the simulation node will be scattered, an input dialog shown in Figure 18 will appear asking the user to fill the width, height of the simulation area and the number of clients that need to be generated.

Figure 18: Simulation area

Servers: This operation will display to the user all server information as mentioned on the database, with the ability to add new server and edit some of the servers attribute (Memory, CPU … etc). Server list window is shown in Figure 19.

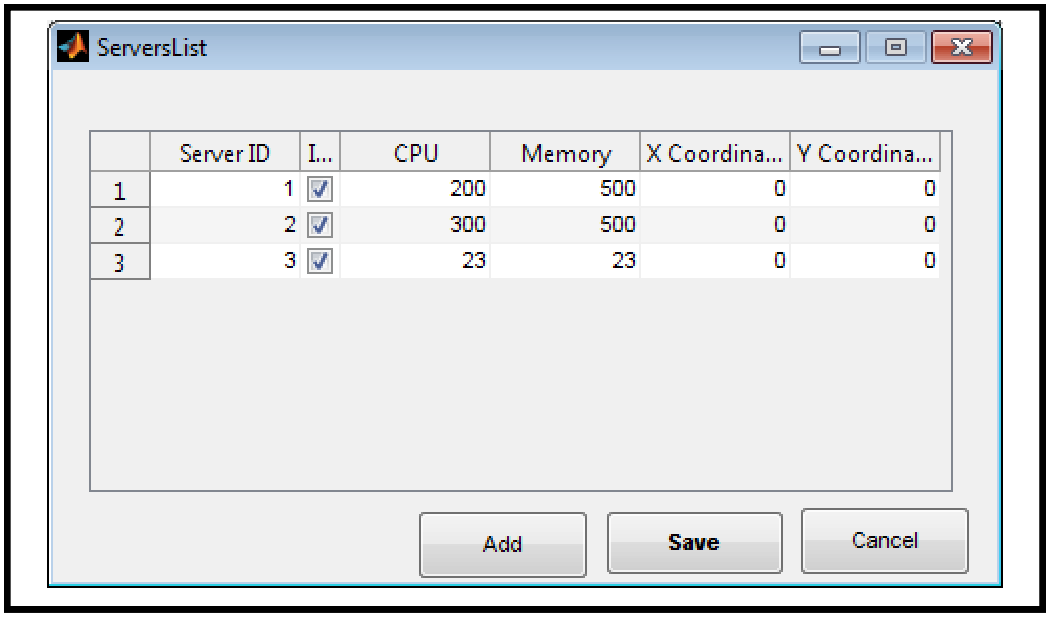
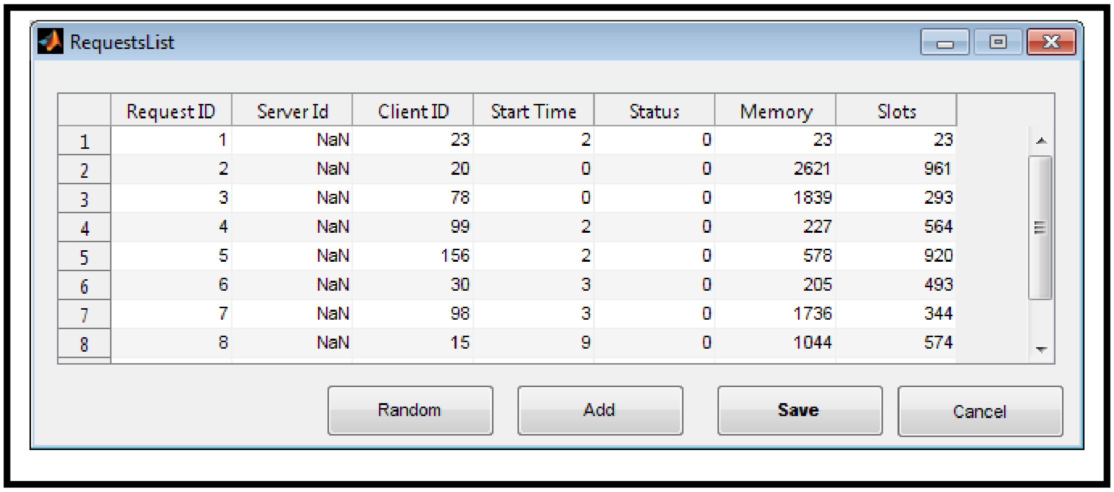


Figure 19: Servers list

Prepare: This operation does the following

1. Randomly scatter the servers and the clients in the simulation area, using the area specified before.
2. Train the ANN after asking the user if s/he would like to train the ANN.
3. Load all servers defined in the database.

Requests: This operation will display the list of the request on the system and all editable attributes which user can modify as described in the database, and enable the user to add new request and randomly generate request. Request list window is shown in Figure 20.



	Request ID	Server Id	Client ID	Start Time	Status	Memory	Slots
1	1	NaN	23	2	0	23	23
2	2	NaN	20	0	0	2621	961
3	3	NaN	78	0	0	1839	293
4	4	NaN	99	2	0	227	564
5	5	NaN	156	2	0	578	920
6	6	NaN	30	3	0	205	493
7	7	NaN	98	3	0	1736	344
8	8	NaN	15	9	0	1044	574

Figure 20: Requests list

In order to let the user to generate requests randomly an input dialog will appear asking him/her to determine attributes for the random process such as (1) minimum and maximum values for the request memory and CPU, (2) number of request to be generated (3) maximum time when the request will be started. Random parameter dialog is shown in Figure 21.

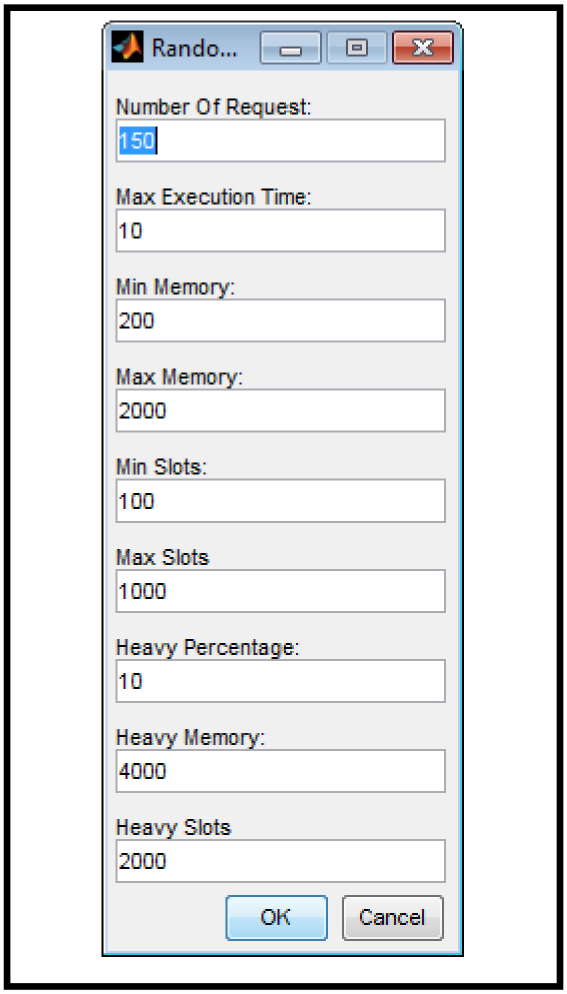


Figure 21: Random parameter

Start Simulation: This operation is responsible for implementing the main server and servers algorithms as described before in chapter 3 and it has the following MATLAB function

```
function [ output_args ] = Proccess( dbConn, newServerList, type, AdditionalArgument )
%PROCCESS Summary of this function goes here
%   Detailed explanation goes here
timer = 0;
serversCount = size(newServerList, 1);

proccessedRequestsSlots = containers.Map();
requestStartTime = containers.Map();
requestStartSlot = containers.Map();

sleepTime = cell2mat(runstoredprocedure(dbConn,'MS_SettingsGetSleepTime',{1}, {java.sql.Types.INTEGER}));
unfishedRequest = cell2mat(runstoredprocedure(dbConn,'MS_RequestUnFinished',{}, {java.sql.Types.INTEGER}));
whileunfishedRequest>= 1
AssignRequest(dbConn, newServerList, timer, type, AdditionalArgument);

pause (sleepTime)

forserverIndex = 1:serversCount
CPUSlots = newServerList(serverIndex, 5) + newServerList(serverIndex, 7);
IntSlots = floor(CPUSlots);
lastIndex = newServerList(serverIndex, 6);
totalSlots = newServerList(serverIndex, 8);
ifCPUSlots>= 1
resultData = cell2mat(runstoredprocedure(dbConn,'MS_RequestsProccessServerSlots',{newServerList(serverIndex, 1), IntSlots, totalSlots, timer, lastIndex}, {java.sql.Types.INTEGER, java.sql.Types.INTEGER}));

while true
if size( resultData, 1) == 2
lastIndex = resultData(1);
ifresultData(2) == IntSlots
break;
else
totalSlots = totalSlots + resultData(2);
IntSlots = IntSlots - resultData(2);
lastIndex = 0;
end
elseiflastIndex == 0
break;
else
lastIndex = 0;
end
```

```
resultData =
cell2mat(runstoredprocedure(dbConn,'MS_RequestsProccessServerSlots',{newServerList(serverIndex, 1), IntSlots, totalSlots, timer, lastIndex}, {java.sql.Types.INTEGER, java.sql.Types.INTEGER}));
end

totalSlots = totalSlots + IntSlots;
end
newServerList(serverIndex, 6) = lastIndex;
newServerList(serverIndex, 7) = CPUSlots - floor(CPUSlots);
newServerList(serverIndex, 8) = totalSlots;
end

timer = timer + 1;
sleepTime = cell2mat(runstoredprocedure(dbConn,'MS_SettingsGetSleepTime',{1}, {java.sql.Types.INTEGER}));
unfishedRequest = cell2mat(runstoredprocedure(dbConn,'MS_RequestUnFinished',{}, {java.sql.Types.INTEGER}));
end
end
```

Appendix B: Artificial Neural Network MATLAB functions

In this appendix we mention some functions that are used at our simulation networks that depend on Artificial Neural Network.

The following function shows the creation of our neural network as well as the technique used for training this newly created neural network:

```
function [ ANN ] = CreateANN( dbConn )
% the function used to create the new feed-forward neural network and
% learning it using the data base connection passed to it as input

% Get the training set from the manual technique which can be found in the
% input database connection
% these record will be randomly selected from database level
cellTraning = fetch(dbConn,'SH_GetANNTraingData');

% Convert the received cell type to number type (integer) since the ANN does not
% accept cell data type
nomaricTraining = cel2mat(cellTraning);
[NumberOrRows NumberOfCols] = sizeof(nomaricTraining);

% avoid using the first two column since they have the request id (which is
% not important) and server id (which will be used for the output or target set)
% we flip the matrix so it can be accepted by the MATLAB neural network
trainingSet = nomaricTraining(:, 3:NumberOfCols)';

% this is custom function used to convert the numeric value to its binary
% presentation
targetSet = Number2Binary(nomaricTraining, 2)';

% creation of our ANN
ANN = newff(minmax(trainingSet),[9 2] ,{ 'tansig'  'logsig' },'trainlm','learngdm','mse');
ANN = init(ANN) ;
ANN.trainparam.epochs = 10000;
ANN.trainparam.goal = 0.01;
ANN = train(ANN, trainingSet, targetSet);

% Compare between the actual and desired output
% this for debugging purpose only
% trainedout = sim(ANN, trainingSet);
% diff = ComparMatrix(targetSet, round(trainedout));
end
```

The second function described how we used the created ANN above for assigning any new request:

```
function ANNSelection( serversList, timer, request, AdditionalArgument )
% this function used to assign the input request to appropriate server using
% the neural networked which should passed as the additinal argument

% First we need to get all servers status (remain time to finish all assigned request,
% available memory ... and so on
currentServersStatus = fetch(dbConn,'SH_GetCurrentServersStatus');

% from the retrieved data we need to build record that is accepted by the
% created neural network in terms of the number of record and their data type
ANNInput = BuildANNInputData(currentServersStatus);

% passes the ANN input to the trained neural networks
serverIDBinaryForm = sim(AdditionalArgument, ANNInput);

% convert the binary presentation of the server is to its corresponding number
serverID = Binary2Numer(serverIDBinaryForm);

[numberOfServers numberOfAtributes] = sizeof(round(currentServersStatus));

% check if the server id is valid
% it is impossible that the server id will not be valid but just to be sure
if serverID < numberOfServers
   % this new request is new ready to be assign to the server id
   AssignRequestToServer(currentServersStatus, serversList, timer, request);
else
   % this error will terminate the execution of the program
   error('invalid server id');
end

end
```

الإستغلال الامثل لمصادر الخادمات المتكررة باستخدام خوارزمية الشبكات العصبية الإصطناعية

إعداد

رضوان محمد رضوان الشلالفه

المشرف

الدكتور موسى توفيق الاخرس

المشرف المشارك

الدكتور عزام طلال سليط

ملخـــــص

استناداً على حقيقة أن هناك كميات ضخمة من البيانات متاحة لأي مستخدم على شبكة الانترنت وهذه البيانات في زيادة مستمرة، هذا يعني ان هناك طلب كبير من مستخدمي الإنترنت على الخوادم لتحميل أو معالجة البيانات، وبالتالي فإن الحاجة إلى زيادة اداء الخوادم اصبح موضوعاً مهماً يجب التصدي له.

ان تكرار الخوادم طريقة لتقليل الضغط الناتج عن زيادة الطلب لمستخدمي الانترنت و زيادة اداء هذه الخوادم، ولكن هناك حاجة أيضا لإعادة توجيه طلب المستخدم لخادم واحد من تلك الخوادم المتكررة، وبذلك فان وجود هذه الخوادم سيكون مجهولا بالنسبة للعديد من مستخدمين الإنترنت وهذا ما يسمى تقنية الخوادم المتكررة. لقد تم اجراء العديد من الدراسات لإيجاد نموذج أو تقنية مناسبة لتتمكن تقنية الخوادم المتكررة من إختيار الخادم الأنسب لأي طلب من مستخدمي الإنترنت.

في هذا البحث، تم استخدام تقنية من تقنيات الذكاء الاصطناعي والتي تسمى بالشبكات العصبية الاصطناعية لإختيار الخادم المناسب (أفضل خادم) لطلب مستخدم جديد، ان الشبكات العصبية الاصطناعية تأخذ مميزات الخوادم الحالية بالاضافة الى طلب المستخدم الجديد كمدخلات، وتعطي الخادم المناسب كمخرج، لكنها بحاجة إلى التدريب لكي نتمكن من تهيئتها لذلك تم استخدام البيانات من تقنية اخرى تسمى الاختيار اليدوي عن طريق الإنسان.

لقد تم مقارنة فعالية الشبكات العصبية مع تقنيتين مختلفتين: اختيار الإنسان و جولة روبن،

ومن هذة المقارنة تم الإستدلال على أن الشبكات العصبية الإصطناعية استطاعت الاستفادة من هاتين التقنيتين والتغلب ايضاً على مساوئها، واستطاعت اعطاء نتائج مشابهة للتقنية اليدوية من حيث الدقة ولكن بآلية إختيار أوتوماتيكية لا تعتمد على تدخل البشر.